gnuplot for TI Graphing Calculator Users

Nathaniel Beaver

August 8, 2013

Contents

0.1 Author's notes . . . 2

1 Introduction **3**
1.1 Running gnuplot interactively . . . 3

2 Representing numbers **6**
2.1 Scientific notation . . . 6
2.2 Predefined constants . . . 7
2.3 Complex numbers . . . 7
2.4 Integer and floating point limits . . . 7
2.5 Summary . . . 7

3 Storing and comparing numbers **8**
3.1 Multiple assignments . . . 8
3.2 Binary comparisons . . . 9

4 Basic arithmetic **11**
4.1 Differences between arithmetic expressions in TI-BASIC and gnuplot . . . 11
4.1.1 Integer arithmetic . . . 11
4.1.2 No implied multiplication . . . 12
4.1.3 Exponentiation . . . 12
4.2 Order of operations . . . 12
4.3 Summary . . . 12

5 Trigonometry, logarithms, and other special functions **14**
5.1 Divisions by zero, NaNs, and other edge cases . . . 15
5.2 Complex arithmetic . . . 16

6 Defining functions **17**

7 Plotting functions in 2D **19**
7.1 Interactively panning the viewing window . . . 21
7.2 Interactive zooming . . . 21
7.3 Plotting display tricks . . . 22
7.4 Modifying plots using the command line . . . 22
7.5 Terminals . . . 22
7.6 Scripting . . . 22
7.7 Dynamic plots . . . 22

8 Bibliography **23**

9 History of gnuplot and TI graphing calculators **24**

0.1 Author's notes

This work also includes a copy of the LaTeX source code used to generate it as a PDF attachment.

`http://help.adobe.com/en_US/acrobat/using/WS58a04a822e3e50102bd615109794195ff-7cb8.w.html#WS58a04a822e3e50102bd615109794195ff-7ca5.w`

`http://blogs.igalia.com/carlosgc/2006/04/06/evince-attachments/`

The inclusion of this code is intended to make it easier to modify and reuse this work.

Chapter 1

Introduction

Much like a TI graphing calculator, gnuplot is primarily for plotting, but can also do basic calculations and statistical functions. What follows is aimed at people most familiar with TI graphing calculators who would like to learn to use gnuplot to accomplish what they do with their graphing calculators.

This is, of course, a very limited subset of what is possible with gnuplot, but I was unable to find a guide explicitly addressing this subset, a guide which I think would be of benefit to me and others. Gnuplot is such a flexible and comprehensive program that it is difficult to know which part to learn first and what to ignore. By focusing on the overlapping functionality of a TI graphing calculator and gnuplot, it becomes much easier to make these choices.

The plan is to start with interactive evaluation and plotting, then move into a little bit of scripting at the end. Gnuplot is a full-blown language like TI-BASIC, but there are plenty of good tutorials and guides to take full advantage of this, so I won't even attempt to fully outline the power of gnuplot's programming features.

I should also note that while gnuplot has a surprisingly large number of features, it is not intended as a general-purpose calculator. For example, if you want to do arbitrary precision floating point calculations or use matrices, gnuplot is a poor choice, and you should look elsewhere. Gnuplot does one thing well: drawing plots.

1.1 Running gnuplot interactively

To run gnuplot interactively, just type `gnuplot` into your shell of choice, or, if you are using Windows, with the provided graphical interface.

```
nathaniel@nathaniel-laptop:~$ gnuplot

        G N U P L O T
        Version 4.4 patchlevel 3
        last modified March 2011
        System: Linux 3.2.0-41-generic-pae

        Copyright (C) 1986-1993, 1998, 2004, 2007-2010
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help␣seeking-assistance"
```

```
        immediate help:   type "help"
        plot window:      hit 'h'

Terminal type set to 'wxt'
gnuplot>
```

Ok, let's try something!

```
gnuplot> 1+1
         ^
         invalid command
gnuplot>
```

Oops! Unlike a TI graphing calculator, gnuplot doesn't assume we want a number when we input an expression. We have to be specific about what we want. For gnuplot, this means using the `print` statement.

```
gnuplot> print 1+1
2
```

Gnuplot has interactive help for all of its commands.

```
gnuplot> help print
 The ‘print‘ command prints the value of <expression> to the screen.  It is
 synonymous with ‘pause 0‘.  <expression> may be anything that ‘gnuplot‘ can
 evaluate that produces a number, or it can be a string.

 Syntax:
       print <expression> {, <expression>, ...}

 See ‘expressions‘.  The output file can be set with ‘set print‘.
```

Gnuplot even has help for the `help` command.

```
gnuplot> help help
 The ‘help‘ command displays built-in help. To specify information on a
 particular topic use the syntax:

       help {<topic>}

 If <topic> is not specified, a short message is printed about ‘gnuplot‘.
 After help for the requested topic is given, a menu of subtopics is given;
 help for a subtopic may be requested by typing its name, extending the help
 request.  After that subtopic has been printed, the request may be extended
 again or you may go back one level to the previous topic.  Eventually, the
 ‘gnuplot‘ command line will return.

 If a question mark (?) is given as the topic, the list of topics currently
 available is printed on the screen.
```

To end an interactive session, you may use the `quit` or `exit` commands, or if you are feeling laconic, `Ctrl-d`, `q`, and `ex` work as well. You can, of course, use the mouse to exit, but gnuplot is optimized for keyboard shortcuts and it is rewarding to those who learn them.

Chapter 2

Representing numbers

Most of the time, if you just want a number, you can simply type it in to gnuplot and evaluate it with the `print` statement.

```
gnuplot> print 1234567890
1234567890
gnuplot> print 0.987654321
0.987654321
```

According to the manual, "any mathematical expression accepted by C, FORTRAN, Pascal, or BASIC is valid." You can get more information with the built-in help.

```
gnuplot> help expressions
```

It is important to be aware that gnuplot distinguishes between integers and floating point numbers, and that these are both distinct from a string representation of a number.

```
gnuplot> print 1.000
1.0
gnuplot> print 1
1
gnuplot> print '1.000'
1.000
```

We will see why this is important once we get to division.

2.1 Scientific notation

gnuplot will accept numbers in a variety of scientific notations.

```
gnuplot> print 6.02e23
6.02e+23
gnuplot> print 6.02E23
6.02e+23
gnuplot> print 6.02e+23
6.02e+23
gnuplot> print 6.02E+23
6.02e+23
gnuplot> print 60.2e22
```

```
6.02e+23
gnuplot> print 602E21
6.02e+23
gnuplot> print 6.67e-11
6.67e-11
gnuplot> print 6.67E-11
6.67e-11
```

2.2 Predefined constants

gnuplot has the trigonometric constant π available as `pi`; unfortunately, the base of the natural logarithms, `e`, is only available as `exp(1)`.

```
gnuplot> print pi
3.14159265358979
gnuplot> print e
         undefined variable: e

gnuplot> print exp(1)
2.71828182845905
```

However, it is possible to overwrite the value of `pi`, so you may wish to use `GPVAL_pi` instead:

```
gnuplot> print GPVAL_pi
3.14159265358979
```

2.3 Complex numbers

gnuplot, much like Fortran or recent versions of C, supports complex numbers.

```
gnuplot> print sqrt(-1)
{0.0, 1.0}
```

However, all complex numbers are floating point; there are no complex integers.

```
gnuplot> print {0,1}
{0.0, 1.0}
```

2.4 Integer and floating point limits

2.5 Summary

Here is a table summarizing the differences between gnuplot and TI calculators that we have discussed so far.

TI calculator	gnuplot syntax	Meaning
EE	`e` or `E`	Scientific notation.
π	`pi`	The mathematical constant, $\approx 3.14159...$
e	`exp(1)`	The mathematical constant, $\approx 2.71828...$
i	`{0,1}`	The square root of -1

Chapter 3

Storing and comparing numbers

Generally, we would like to store values for later. This is quite easy to do in gnuplot, and you can name a variable anything you like, provided it starts with a letter and contains letters, numbers, and _ (an underscore).

```
gnuplot> my_variable = 3
gnuplot> print my_variable
3
```

If you want to know a variable's value, use

```
print my_variable
```

This is equivalent to

```
print value("my_variable")
```

If you are only interested in whether or not it is defined, use

```
print exists("my_variable")
```

If you want to delete a variable's value, use

```
undefine my_variable
```

3.1 Multiple assignments

gnuplot is written in C and takes many of its syntactic cues from C. For example, if you want to make multiple assignments on a single line, use the semicolon ; to indicate the end of a statement.

```
gnuplot> x = 3 y = 4 z = 5
                 ^
         ';' expected

gnuplot> x = 3; y = 4; z = 5
```

Semicolons for single statements are mandatory in C, but optional in gnuplot.

```
gnuplot> a = 3
gnuplot> a = 3;
```

Finally, you can use semicolons to join any multi-line statements, not just assignments.

```
gnuplot> print x, y, z
3 4 5
gnuplot> print x; print y; print z
```

```
3
4
5
```

3.2 Binary comparisons

We can also compare values and variables using the six binary comparisons:

```
gnuplot> print 1 == 2
0
gnuplot> print 1 == 1
1
gnuplot> print 1 < 2
1
gnuplot> print 1 > 2
0
gnuplot> a = 3
gnuplot> print a == 3
1
gnuplot> print a > 3
0
gnuplot> b = 4
gnuplot> print a > b
0
gnuplot> print a < b
1
gnuplot> print a >= b
0
gnuplot> print a <= b
1
gnuplot> print a != b
1
```

gnuplot is very accommodating. If, for example, you wish to redefine[1] the value of π, gnuplot will happily follow along.

```
gnuplot> pi = 3.2
gnuplot> print pi
3.2
```

You can also redefine the value of `NaN` (not a number) so that it is a number.

```
gnuplot> NaN = 4
gnuplot> print NaN
4
```

If you would like to see what variables you have defined, you can use a new command, `show`.

```
gnuplot> show variables
```

[1]The Indiana legislature had a bill to do just that in 1897, and nobody has let them forget since. `http://www.agecon.purdue.edu/crd/Localgov/Second%20Level%20pages/indiana_pi_bill.htm`

```
        User and default variables:
        pi = 3.14159265358979
        NaN = NaN
        GNUTERM = "wxt"
        a = 3
        b = 4
```

The built-in values `GPVAL_pi` and `GPVAL_NaN` cannot be modified, however.

```
gnuplot> GPVAL_pi = 3.2
         ^
         Cannot set internal variables GPVAL_ and MOUSE_
gnuplot> GPVAL_NaN = 2
         ^
         Cannot set internal variables GPVAL_ and MOUSE_
```

You can see these variables with `show variables all` or `show variables GPVAL_`.

```
gnuplot> show variables GPVAL


        Variables beginning with GPVAL:
        GPVAL_TERM = "wxt"
        GPVAL_TERMOPTIONS = "0"
        GPVAL_OUTPUT = ""
        GPVAL_VERSION = 4.4
        GPVAL_PATCHLEVEL = "3"
        ...(snip)
        GPVAL_pi = 3.14159265358979
        GPVAL_NaN = NaN
        GPVAL_ERRNO = 1
        GPVAL_ERRMSG = "invalid␣command"
        GPVAL_PWD = "/home/nathaniel"
```

Here's a summary of storing and comparing variables.

TI calculator	gnuplot syntax	Meaning
`3→A`	`A=3`	Store the value 3 into the variable A.
`Disp A`	`print A`	See the value of the variable A.
`expr("A")`	`value("A")`	Get the value of the variable A from the string "A".
(none)	`exists("A")`	Check if the variable `A` is defined.
`DelVar A`	`undefine A`	Clear the value of the variable `A`.
$=$	`==`	Equality of expressions.
$\neq$	`!=`	Inequality of expressions.
$>$	`>`	Greater than.
$<$	`<`	Less than.
$\geq$	`>=`	Greater than or equal to.
$\leq$	`<=`	Less than or equal to.
:	;	Multiple statement separator.

Chapter 4

Basic arithmetic

gnuplot can do interactive arithmetic just fine, although it has a few quirks.

```
gnuplot> print 1+2
3
gnuplot> print 2*3
6
gnuplot> print 3-4
-1
```

4.1 Differences between arithmetic expressions in TI-BASIC and gnuplot

4.1.1 Integer arithmetic

gnuplot defaults to integer arithmetic only, so you must explicitly declare a floating point like 1.0/3

```
gnuplot> print 1/3
0
gnuplot> print 1.0/3
0.333333333333333
```

If you want to guarantee that a value is floating point before dividing it by something, just add 0.0 or multiple by 1.0.

```
gnuplot> a = -2
gnuplot> b = -3
gnuplot> print a/b
0
gnuplot> print b/a
1
gnuplot> print (b + 0.0)/a
1.5
gnuplot> print (b * 1.0)/a
1.5
```

4.1.2 No implied multiplication

If you want gnuplot to multiply something, you must tell it explicitly. If you don't, it will think you are doing multiple statements on a single line and forgot a semicolon.

```
gnuplot> print 3(4+5)
                ^
         ';' expected

gnuplot> print 3*(4+5)
27
```

4.1.3 Exponentiation

gnuplot uses ** instead of ^ for exponentiation:

```
gnuplot> print 3**4
81
gnuplot> print 3**(0.25)
1.31607401295249
```

This does take some getting used to. If it is too problematic to adjust to this, you may want to use different mathematical software, such as Maxima or Octave, instead. These can call gnuplot for the plotting routines, and do all the input and number-crunching by themselves.

4.2 Order of operations

A lot of manuals like to talk about order of operations, because they are important if you are designing a language or can't bear to use disambiguating parentheses.

However, if it's not obvious what the order of operations is, do not depend on language-dependent rules. Just add parentheses to clarify for whoever is trying to read your code, including yourself.

```
gnuplot> x = 3
gnuplot> print 1/2*x
0
gnuplot> print 1.0/2.0*x
1.5
gnuplot> print 1.0/(2.0*x)
0.166666666666667
gnuplot> print (1.0/2.0)*x
1.5
```

4.3 Summary

Here is a table showing the difference between the key you would press on a TI calculator and gnuplot syntax for basic arithmetic, exponentiation, and multi-line statements.

TI calculator	gnuplot syntax	Meaning
$+$	$+$	Addition or unary positive sign, e.g. $3+4=7$, $+3$.
(-)	`-`	Unary minus sign, e.g. -3.
$-$	`-`	Subtraction, e.g. $3-4=-1$.
$\times$	$*$	Multiplication, e.g. $3\times 4=12$.
$\div$	$/$	Division, e.g. $\frac{1}{2}=0.5$.
x^2	`x**2`	Squaring, e.g. $3^2=9$.
$\sqrt{(}$	`sqrt()`	Square root, e.g. $\sqrt{4}=2$.
x^{-1}	`1/x` or `x**-1`	Inverse, e.g. $2^{-1}=0.5$.
e^x	`exp()`	Exponential function, e.g. $e^1=exp(1)=2.71828...$
10^x	`10**x`	Power of ten, e.g. $10^2=100$.

Chapter 5

Trigonometry, logarithms, and other special functions

gnuplot has the same trig and log functions you can find in the C standard library or Fortran. It also has a few more (e.g. Bessel functions, elliptic integrals, and variations on the gamma function), but we will only bother with the ones available on a TI calculator. You can see all available built-in functions with `help functions`.

gnuplot defaults to radians, but you can switch to degrees.

```
gnuplot> show angles

        Angles are in radians

gnuplot> print cos(pi)
-1.0
gnuplot> set angles degrees
gnuplot> print cos(180)
-1.0
```

This is a summary of logarithmic and trigonometric functions and their syntax.

TI calculator	gnuplot syntax	Domain (real result)	Domain (complex result)	Range
log	`log10`	$(0, \infty)$	$(-\infty, 0)$	$(-\infty, \infty)$
ln	`log`	$(0, \infty)$	$(-\infty, 0)$	$(-\infty, \infty)$
sin	sin	$(-\infty, \infty)$	$\varnothing$	$[-1, 1]$
cos	cos	$(-\infty, \infty)$	$\varnothing$	$[-1, 1]$
tan	tan	$x \neq (n + 1/2)\pi$		$(-\infty, \infty)$
$\sin^{-1}$	asin	$[-1, 1]$		
$\cos^{-1}$	acos	$[-1, 1]$		
$\tan^{-1}$	atan	$(-\infty, \infty)$		
sinh	sinh			
cosh	cosh			
tanh	tanh			
$\sinh^{-1}$	asinh			
$\cosh^{-1}$	acosh			
$\tanh^{-1}$	atanh			

5.1 Divisions by zero, NaNs, and other edge cases

If you try to divide by zero or take the log of zero, gnuplot will complain.

```
gnuplot> print 1/0
               ^
         undefined value

gnuplot> print 0/0
               ^
         undefined value

gnuplot> print log(0)
               ^
         undefined value
```

Similarly, trying to evaluate too large a number will throw an error. Remember when $69! \approx 1.711 \times 10^{98}$ was as high as we could go?

```
gnuplot> print 170!
7.257415615308e+306

gnuplot> print 171!
               ^
         undefined value
```

However, trig functions suffer from rounding error, such that they might not always behave as you would expect.

```
gnuplot> print tan(pi/2)
```

```
1.63317787283838e+16
gnuplot> print tan(3.14159265358979323846264338327950288419716939937/2)
1.63317787283838e+16
gnuplot> print tan(3.1416/2)
-272241.808409276
```

On the bright side, gnuplot will give you complex numbers when they are the well-defined results of a special function.

```
gnuplot> print sqrt(-4)
{0.0, 2.0}
gnuplot> print log(-1)
{0.0, 3.14159265358979}
gnuplot> print asin(-2)
{-1.5707963267949, -1.31695789692482}
```

5.2 Complex arithmetic

Chapter 6

Defining functions

Until now, we have been using gnuplot as a glorified desk calculator, a very limited area of its capabilities. It is time we starting defining functions we want to plot.

Those familiar with functions in other programming languages may find gnuplot functions too restrictive; those familiar with TI calculator functions will be right at home.

Functions take one or more values (a.k.a. arguments or parameters) as input and return a single value.

```
gnuplot> f(x) = x**2
gnuplot> print f(3)
9
```

I regret to inform you that gnuplot limits you to 12 arguments, so anyone expecting to work with 13 dimensional spaces will need to find another approach.

```
gnuplot> f(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12) = 4
gnuplot> f(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13) = 4
                                                        ^
         function contains too many parameters
```

Functional programming fans will be happy to hear that gnuplot allows recursive functions. You can define the base case using the ternary operator `?`, familiar to C fans.

```
gnuplot> f(x) = x == 0 ? 1 : x * f(x-1)
gnuplot> print f(0), f(1), f(2), f(3), f(4)
1 1 2 6 24
```

As always, though, beware of stack overflows and limited recursion depth.

```
gnuplot> print f(0.1)
         stack overflow

gnuplot> print f(98)
9.42689044888324e+153
gnuplot> print 98!
9.42689044888325e+153
gnuplot> print f(99)
```

```
          stack overflow

gnuplot> print 99!
9.33262154439442e+155
gnuplot> f(x) = f(x)
gnuplot> print f(0)
          recursion depth limit exceeded
```

We ought to address one aspect of computer languages known as scope. Gnuplot has very simple scoping rules. For example, function parameters always trump ordinary variables.

```
gnuplot> a = 3
gnuplot> myfunc(a) = a
gnuplot> print myfunc(4)
4
```

Functions and variables have different namespaces, so they do not conflict. It is generally not necessary (and probably confusing) to have a variable and a function with the same name, but you can if you want to.

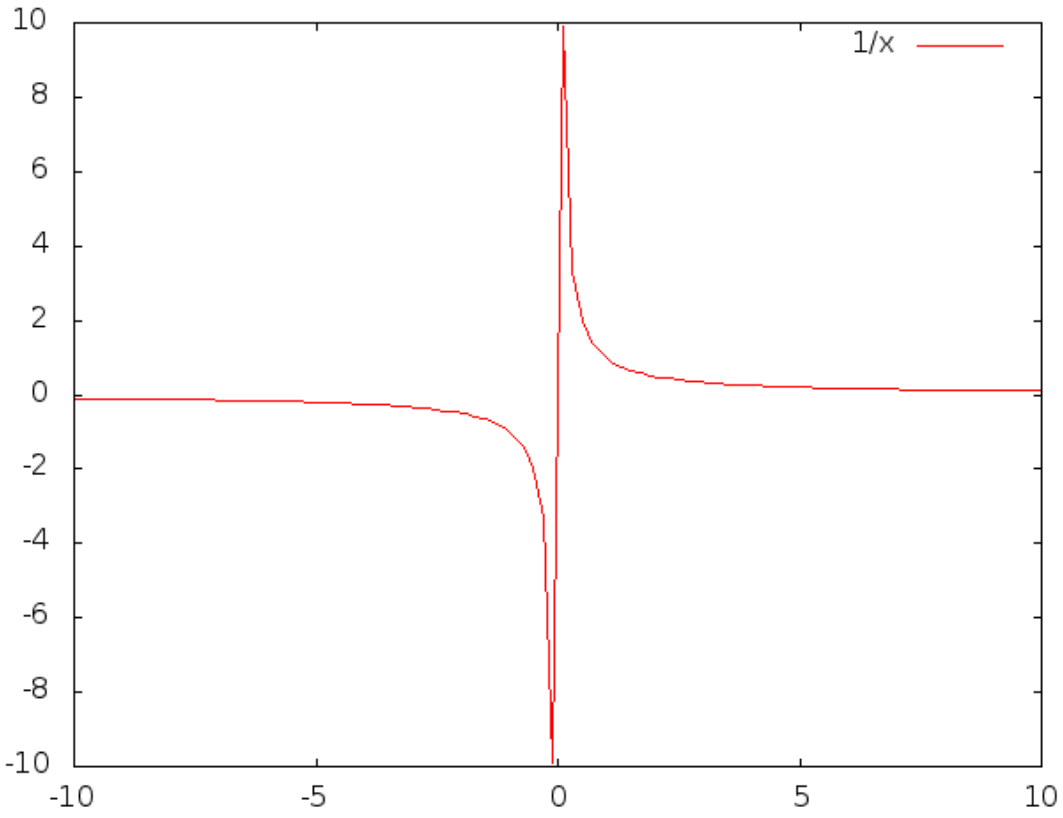
```
gnuplot> a = 3
gnuplot> a(a) = a
gnuplot> print a(4)
4
```

Chapter 7

Plotting functions in 2D

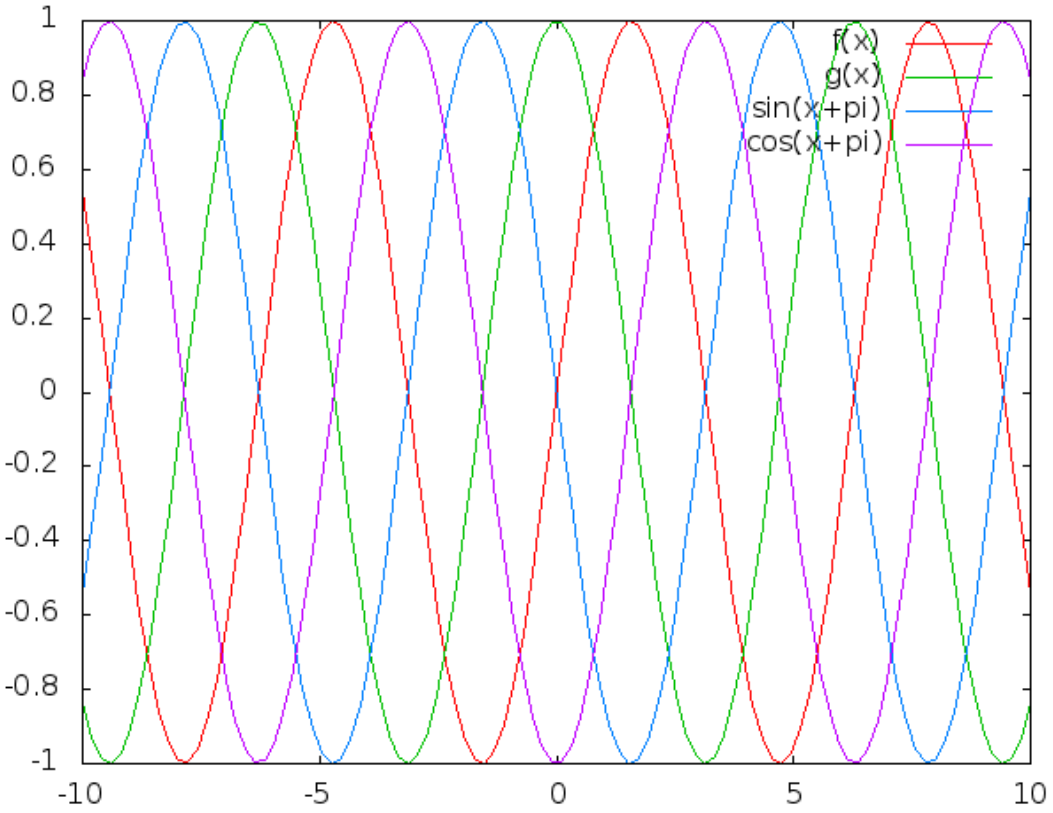
Now that we know how to define functions, it is high time we plot them.

```
gnuplot> plot 1/x
```



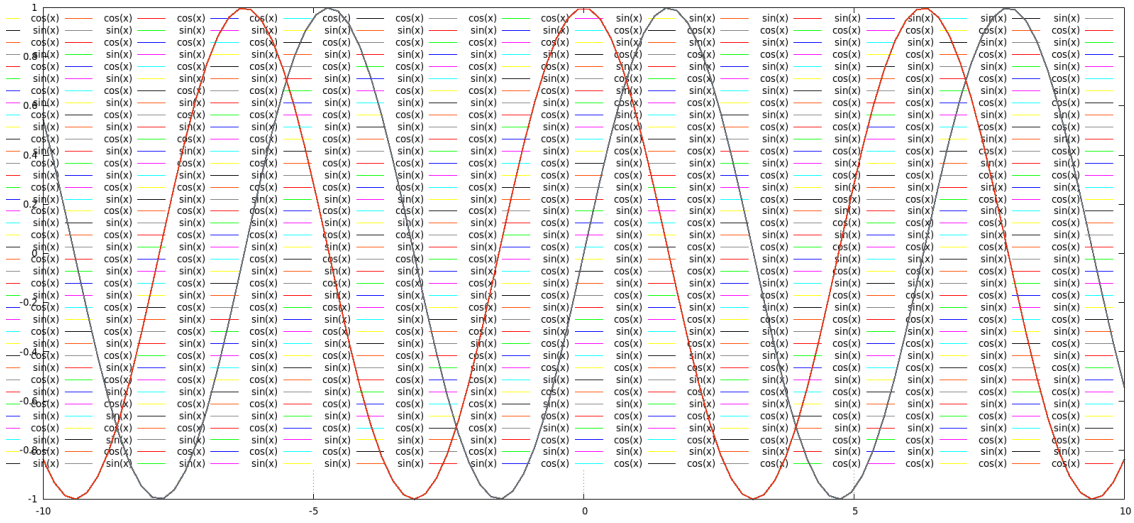
You may plot as many functions as you like, user-defined or built-in.

```
gnuplot> f(u)=sin(u)
gnuplot> g(t)=cos(t)
gnuplot> plot f(x),g(x)
gnuplot> plot f(x),g(x),sin(x+pi),cos(x+pi)
```



I really do mean as many as you like; you are only limited by your patience and the memory of your computer.

```
gnuplot> plot sin(x), cos(x), sin(x), cos(x), sin(x), cos(x), sin(x) ...
```



You can usually match a function to its graph by the color, but gnuplot also provides different plotting styles.

You may have noticed that we can use any independent variable we feel like when defining a function, but we always use `x` when plotting. In fact, if we try to use another variable, we get an error.

```
gnuplot> f(t) = t**2
gnuplot> plot f(t)
         undefined variable: t
```

This is because `x` and `y` are special variables gnuplot calls *dummy variables*.

```
gnuplot> show dummy

dummy variables are "x" and "y"
```

You can change these with `set dummy`. For details, see `help dummy`.

7.1 Interactively panning the viewing window

The default plotting window in gnuplot, `wxt`, has these options for moving around.

Command	Effect
Wheel up	Pan up
Wheel up	Pan down
Shift + wheel up	Pan left
Shift + wheel down	Pan right

7.2 Interactive zooming

Gnuplot has a good autoscaling algorithm, so many of the plotting options in a TI Calculator are unnecessary. Furthermore, there are many options for using mouse and keyboard in the plotting window. To see them, type `h` while in a plotting window. In the documentation, these are the meanings for various mouse actions:

`<B1>`	left click
`<B1-Motion>`	left click and drag
`<B2>`	middle click
`<B2-Motion>`	scroll wheel
`<B3>`	right click

Here are some of the most useful commands for 2D zooming.

Command	Meaning
Right-click and release	Choose zoom window
Ctrl + Scroll up	Zoom in
Ctrl + Scroll down	Zoom out
`q`	Exit plot and return to command line
`e`	Replot
`a`	Autoscale
`p` and `n`	Hop back and forwards to previous and next zoom settings
`u`	Unzoom (reset to original plot settings)
`7`	Cycle through aspect ratios

Some of the TI auto-zoom functions do transfer to gnuplot; others do not exist as built-in options.

TI calculator	gnuplot command	Meaning
`Zbox`	Right-click	Select a box with the cursor and zoom to it.
`Zoom In`	Ctrl-Wheel-Up	
`Zoom Out`	Ctrl-Wheel-Down	
`Zdecimal`	No equivalent	
`Zsquare`	`set size square`	
`Zstandard`	`set size square`	
`Ztrig`	`set size square`	
`Zinteger`	`set size square`	
`Zinteger`	`set size square`	
`Zinteger`	`set size square`	

7.3 Plotting display tricks

Command	Effect
Double left-click	Copy coordinates to clip board
Command	Meaning
Command	Meaning

7.4 Modifying plots using the command line

Gnuplot was used long before computer mouses were widely available, so while the mouse is indubitably convenient, it is not really necessary; anything that can be done with the mouse can be done with the keyboard.

This is actually quite useful for doing things like dynamics plots and scripting, so it's worth learning even if you don't use it much in practice.

7.5 Terminals

7.6 Scripting

7.7 Dynamic plots

```
gnuplot> plot 1/0
                 ^
         all points y value undefined!


nathaniel@nathaniel-laptop:~$ gnuplot -e "print rand(0)"
0.222457440974512
nathaniel@nathaniel-laptop:~$ gnuplot -e "print rand(0)"
0.222457440974512
```

Chapter 8

Bibliography

gnuplot 4.7
An Interactive Plotting Program
Thomas Williams & Colin Kelley
2012 Version 4.7 (cvs)

Gnuplot in Action
Understanding Data with Graphs
Philipp K. Janert
2010 by Manning Publications Co.

TI-84 Plus and
TI-84 Plus Silver Edition
Guidebook
2004--2010 Texas Instruments Incorporated

Chapter 9

History of gnuplot and TI graphing calculators